This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the ICC 2007 proceedings.

Preventing Bandwidth Abuse at the Router through Sending Rate Estimate-based Active Queue Management

Venkatesh Ramaswamy, Leticia Cuéllar, Stephan Eidenbenz and Nicolas Hengartner
Computers and Computational Sciences
Los Alamos National Laboratory
PO Box 1663, MS-M997
Los Alamos, NM-87545
Email: {vramaswa,leticia,eidenben,nickh}@lanl.gov

***Abstract*— We propose a rigorous mathematical interpretation of a novel family of Active Queue Management schemes, called Sending Rate Estimate based Queue Management (SREQM) scheme, that aims to provide fair bandwidth allocation to all the flows in a router by estimating the flow sending rates, while maintaining only minimal per-flow state information. We propose an optimized implementation of SREQM, called Fair Sending Rate Estimate based Queue Management (FSREQM) scheme, and show through comparative simulation that FRESQM is the only scheme among those tested that successfully prevents bandwidth abuse while maintaining high link utilization.**

I. Introduction

The Internet has grown exponentially over the past 25 years from a small experimental network to a platform that changed the way we communicate and do business. While from the end user's perspective, the Internet is a shared resource, competition and commercialization have lead to resource abuse. There are a lot of forms of resource abuse, and the most important ones are over usage of link bandwidth and router buffer. We call a situation in which a user can steal all the available bandwidth, starving other users, as "bandwidth abuse".

Bandwidth abuse in the Internet can happen intentionally or unintentionally based on the nature of traffic source. Traffic sources in the Internet are broadly classified as responsive or unresponsive sources based on their reaction to congestion in the network [1]. A source that employs a protocol that respond to congestion by reducing the offered load is called a responsive source. The most common example is a source employing TCP [2]. On the other hand a source with a protocol that ignores congestion by simply maintaining or even increasing its load is termed a non-responsive source, the most common example of which is a source using UDP [2].

The most appropriate response, socially, to congestion, for each source is to reduce the offered load to match the available capacity of the network. Application designers usually choose protocols that best balance quality of service (QoS) requirements of the application and the overall network performance. Applications like file transfer usually adopt a TCP-friendly protocol because integrity of the data is required, while applications such as VoIP usually adopt a non-responsive protocol as delay requirements are high for these applications. While some applications are non-responsive because they have to be, some others are unnecessarily non-responsive.

With the exponential growth of the Internet, there are many instances where the sources can be non-responsive. Even though most of the variations of the TCP implement congestion control, their response to congestion varies. Users can adopt aggressive settings to get more bandwidth. An example would be to use a modified form of TCP with larger initial window and window opening constants. Also, with a large number of short flows, the aggregate traffic may become non-responsive even though each of these short flows is responsive to congestion [3]. Moreover, many growing applications such as packet video and packet audio employ the UDP protocol, which does not implement congestion control. In short, the non-responsive behavior is growing in the Internet.

These non-responsive sources can monopolize network bandwidth and starve the "congestion friendly" flows. Without implementing a centralized congestion control algorithm and forcing the users to abide by that algorithm, it is not possible to guarantee that they will not act in a selfish manner [4]. For large networks such as the Internet, however, forcing all the end users to adopt a centralized algorithm is not feasible. Local congestion control schemes are the only viable approach to prevent bandwidth abuse.

Some congestion control approaches at the router have implicit capability to prevent bandwidth abuse. Congestion control approaches at the router can be mainly classified in to two categories: user-centric congestion control algorithms and router-centric congestion control algorithms [5]. In the user-centric approach the output port of the router maintains a separate queue for packets from each input port. Scheduling algorithms employing user centric approach, of which the Generalized Processor Sharing (GPS), and the Weighted Fair Queueing (WFQ) algorithm are the most generic ones, require the buffer at each output be divided into separate queues to hold the packets from each separate flow [1]. Packets from these queues are then scheduled in a specific way, typically on a round-robin basis. Since packets from each flow are sepa-

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
JUN 2007		**00-00-2007 to 00-00-2007**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Preventing Bandwidth Abuse at the Router through Sending Rate Estimate-based Active Queue Management	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Los Alamos National Laboratory,Computers and Computational Sciences,PO Box 1663, MS-M997,Los Alamos,NM,87545**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADM002055. Proceedings of the 2007 IEEE International Conference of Communications (ICC 2007) Held in Glasgow, Scotland on June 24-28, 2007. U.S. Government or Federal Rights License

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**6**	

rated from one another, a misbehaving flow cannot degrade the quality of a legitimate flow. However, these approaches usually require complicated per flow state information, thus making it too expensive to be widely deployed when the number of flows is large or when the switching speeds are very high [6]. Simply put, the user-centric approach does not scale. Also this approach is unnecessarily complex because most of the flows in the Internet are short flows usually termed as "web mice" [7]. Moreover, having a separate queue for each flow would reduce statistical multiplexing of buffer space [8].

A scheme that tries to approximate WFQ is Stochastic Fair Queueing (SFQ) [9], but it still requires around 2000 queues in a typical network to approach WFQ's performance [9], [5]. A recent scheme called Core Stateless Fair Queueing [4] tries to reduce the state information the routers have to carry in order to prevent misbehavior and provide fairness among flows. The core router's design complexity is reduced considerably, but the total design is still complex because the algorithm requires to extract some information from the packet headers in a different way than the current way of extracting information from the packets.

A router centric approach on the other hand has a very simple design because it does not need to maintain full state information for each flow. Each output port will have a single FIFO queue to hold packets coming from each of the flows. Droptail queueing scheme and the Random Early Drop (RED) algorithm are two of the most citied queue management algorithms using the router centric approach [5]. Queue management schemes that dynamically signal congestion to sources are usually referred to as Active Queue Management (AQM) schemes. The objective of this work is to design congestion control schemes with the following desired properties:

- The congestion control scheme should not rely on the cooperation of the users. The scheme should assume that there will be some sources which behave selfishly.
- The scheme should have minimum complexity; it should have minimum per flow state information, and should be scalable. The scheme should use the available resources (link and buffer) efficiently.
- The scheme should be capable of imposing penalty to misbehaving sources. It should be able to protect the "good" flows from the "bad" flows.

We introduce a class of queue management schemes called Sending Rate Estimate based Queue Management (SREQM) schemes in [1], which introduces a paradigm shift in the packet dropping mechanism. SREQM bases the packet dropping decision on both the estimated sending rate of flows and the queue length instead of just the average queue length. In this work, we focus on presenting a rigorous mathematical interpretation of SREQM along with the introduction of a simple, highly efficient light-weight algorithm called Fair Sending Rate Estimate based Queue Management (FSREQM) scheme. FSREQM can punish misbehaving flows and guard well behaved flows, guaranteeing a certain QoS among flows.

The rest of the paper continues as follows. In the next section we present a rigorous analytic approach to estimate the relative sending rate of flows. Development of a light-weight queue management scheme called Fair Sending Rate based Queue Management scheme based on the relative sending rate estimate follows. Some simulation results to demonstrate the effectiveness of the scheme in preventing bandwidth abuse is presented in Section IV. With a section on conclusion and some future directions, we conclude this paper.

II. Conceptual design of relative sending rate estimator

Most of the queue management schemes at the router either cannot effectively prevent malicious behavior from sources or cannot scale. We believe that the key to prevent malicious behavior is to base the packet dropping decision from flows on the characteristics of each individual traffic flow. In this section, we present a light-weight approach to estimate the relative sending rate of flows with high accuracy.

Assume that time is divided into discrete time slots and that there is always one packet in each time slot. Each flow has a unique identifier given by the four tuple : (source address, source port, destination address, destination port). Because the sending rate of a flow is a non-stationary process, we may want to use a moving average of the proportion of packets from each flow to estimate the instantaneous sending rate of a flow. To compute this estimate, we need to keep in memory a history of all the packets that arrived during a time window, say T time slots. Denote $\tilde{\chi}_j(t)$ to be the indicator function that the packet at time t is from flow j. That is

$$\tilde{\chi}_j(t) = \begin{cases} 1 & \text{if the packet at time slot } t \text{ is from flow } j \\ 0 & \text{otherwise.} \end{cases}$$

Let $\tilde{A}_j(t_1, t_2)$ be the proportion of packets from flow j in $[t_1, t_2]$. For any time interval of length T, say $[t+1-T, t]$, we can write,

$$\tilde{A}_j(t+1-T, t) = \frac{1}{T} \sum_{l=t+1-T}^{t} \tilde{\chi}_j(l). \tag{1}$$

The proportion of from flow j in the time window T can also be recursively written as

$$\begin{aligned} \tilde{A}_j(t+1-T, t) &= \tilde{A}_j(t-T, t-1) \\ &+ \frac{1}{T}\tilde{\chi}_j(t) - \frac{1}{T}\tilde{\chi}_j(t-T). \end{aligned} \tag{2}$$

The above equation says that in order to compute $\tilde{A}_j(t+1-T, t)$, we add the information of the packet that arrived at time t to $\tilde{A}_j(t-T, t-1)$ and delete the information of the oldest packet in the window, which is the packet that arrived at time slot $(t-T)$. The idea is illustrated in Figure 1. In Figure 1 a packet from flow i is indicated as i. The window size and the number of flows is assumed to be seven and three respectively. At time slot 8, a packet from flow 3 arrives and $\tilde{A}_1(2, 8)$ can be computed as

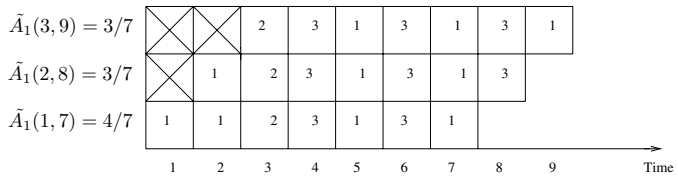

Fig. 1. An example to compute $\tilde{A}_1(\cdot)$

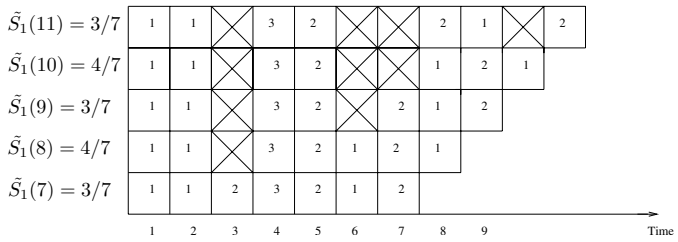

Fig. 2. An example to compute $\tilde{S}_1(\cdot)$

$$\tilde{A}_1(2,8) \quad = \quad \tilde{A}_1(1,7) + \frac{1}{7}\tilde{\chi}_1(8) - \frac{1}{7}\tilde{\chi}_1(1) = 3/7 \quad (3)$$

The above approach requires us to keep track of all the packets that arrived during the last T time slots. That is we need to know the order of last T packet arrivals.

Alternatively, to reduce the computational and memory requirements, we can delete any one of the T packets in the history instead of deleting the oldest packet from history. This only requires us to know the number of packets from each flow in the last T time slots instead of the entire history. If we did that, our estimate would take a form as given below.

$$\tilde{S}_j(t) = \tilde{S}_j(t-1) + \frac{1}{T}\tilde{\chi}_j(t) - \frac{1}{T}\tilde{\Theta}_j(t), \quad (4)$$

where $\tilde{S}_j(t)$ is a new estimate of the relative sending rate of flow j at time slot t and $\tilde{\Theta}_j(t)$ is given as follows

$$\tilde{\Theta}_j(t) = \begin{cases} 1 & \text{if the randomly picked packet from the} \\ & \text{T packets in the history belonged to flow } j \\ 0 & \text{otherwise.} \end{cases}$$

Figure 2 illustrates this idea. As before, we assume 3 flows and the window size as seven. At time slot 8, a packet from flow 1 arrives, but instead of deleting the packet that arrived at time slot one, we randomly pick a time slot and delete the packet that arrived at that time slot. In this specific example, the randomly picked packet turned out to be the one that arrived at time slot three. The packet that arrived at time slot three was from flow 2, and therefore $\tilde{\Theta}_1(8) = 0$. Note that there is always T packets in history at any time. At time slot 8, we can estimate the sending rate of flow 1 as

$$\begin{aligned} \tilde{S}_1(8) &= \tilde{S}_1(7) + \frac{1}{7}\tilde{\chi}_1(8) - \frac{1}{7}\tilde{\Theta}_1(8) \\ &= 3/7 + 1/7 - 0/7 = 4/7. \end{aligned} \quad (5)$$

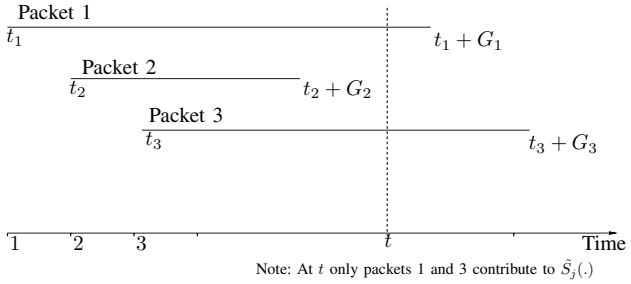

Fig. 3. An alternate way to compute $\tilde{S}_j(t)$

To elucidate the distribution of $\tilde{S}_j(t)$, imagine that the packets from flow j is a realization from a Bernoulli process with parameter P_j. We can show that $\tilde{S}_j(t)$ is an inhomogeneous birth death process that has a Binomial stationary distribution with parameters (T, P_j). As a result, the expected value in the long run is $\lim_{t\to\infty} \mathbf{E}\left[\tilde{S}_j(t)\right] = P_j$. In other words, $\tilde{S}_j(t)$ is essentially an unbiased estimate for the relative frequency of packets from each flow.

By construction of $\tilde{S}_j(t)$, the deletion of packets from memory is random. Therefore, the same arrival history can lead to different realizations of $\tilde{S}_j(t)$. Ideally, we want to remove this randomization by taking the conditional expectation of $\tilde{S}_j(t)$ given the arrival process. That is

$$H_j(t) \quad = \quad \mathbf{E}\left[\tilde{S}_j(t)|\text{history}\right]. \quad (6)$$

Note that

$$\mathbf{E}\left[H_j(t)\right] \quad = \quad \mathbf{E}\left[\mathbf{E}\left[\tilde{S}_j(t)|\text{history}\right]\right] = \mathbf{E}\left[\tilde{S}_j(t)\right] = P_j,$$

and therefore, it can be shown that

$$\mathbf{E}\left[\left(H_j(t) - P_j\right)^2\right] \leq \mathbf{E}\left[\left(\tilde{S}_j(t) - P_j\right)^2\right]. \quad (7)$$

In other words $H_j(\cdot)$ is a tighter estimate for P_j than $\tilde{S}_j(\cdot)$, making $H_j(\cdot)$ a better estimate for P_j than $\tilde{S}_j(.)$.

In order to compute the R.H.S. of Equation (6), we derive an alternate representation for $\tilde{S}_j(t)$ by noting that the probability of picking a packet from T packets in history is $1/T$, and the number of time slots required before a packet is deleted from history is geometrically distributed with parameter $1/T$. We can compute $\tilde{S}_j(t)$ alternatively as follows. When a packet arrives, stamp the packet with a number which is the sum of the time at which it arrives and a number which is drawn from a geometric distribution with parameter $1/T$, which is represented as G_i for packet i. At time slot t, $\tilde{S}_j(t)$ is the proportion of packets from flow j among all the packets whose stamp exceeds t. Figure 3 illustrates the idea. At time slot t, we look at all the packets whose life time exceeds t, that is geometrically distributed with parameter T, and are from flow j. We then divide that by the total number of packets in history which is T. Let $\tilde{G}_i$ be a geometric random variable with parameter $\frac{1}{T}$, $\tilde{S}_j(t)$ can then be represented as

$$\tilde{S}_j(t) = \frac{1}{T}\sum_{l\leq t} \tilde{\chi}_j(l)\tilde{\phi}_l(t) \quad (8)$$

where

$$\tilde{\phi}_l(t) = \begin{cases} 1 & \text{if } \left(\tilde{G}_l + l\right) \geq t \\ 0 & \text{if } \left(\tilde{G}_l + l\right) < t \end{cases}$$

With the above alternate definition of $\tilde{S}_j(t)$, we can compute the R.H.S of Equation (6) as follows

$$\begin{aligned} H_j(t) &= \mathbf{E}\left[\tilde{S}_j(t)|\text{history}\right] = \frac{1}{T}\left(\sum_{l\leq t}\tilde{\chi}_j(l)\right)\mathbf{E}\left[\tilde{\phi}_l(t)\right] \\ &= \frac{1}{T}\sum_{l\leq t}\tilde{\chi}_j(l)\left(1-\frac{1}{T}\right)^{t-l}. \end{aligned} \quad (9)$$

From the above equation, it is readily seen that $H_j(t)$ has a recursive form. $H_j(t)$ can be re-written as

$$H_j(t) = \left(1-\frac{1}{T}\right)H_j(t-1) + \frac{1}{T}\tilde{\chi}_j(t). \quad (10)$$

Both $\tilde{A}_j(\cdot)$ and $H_j(\cdot)$ are estimates of the sending rate of flow j. While $\tilde{A}_j(.)$ is a moving average, $H_j(.)$ is an exponential smoother, where the more recent observations are weighted more than the older observations. Moreover, in $H_j(.)$ we do not have to keep track of the arrival times of the packet as we have to do in $\tilde{A}_j(\cdot)$, and $H_j(\cdot)$ is less variable than $\tilde{S}_j(\cdot)$.

These properties make $H_j(\cdot)$ an obvious choice for estimating the sending rate. Figure 6 illustrates the averaging methods of both the estimates. It can be shown that $\sum_{\forall j} H_j(t) = 1$, which conforms to the fact that $H_j(\cdot)$ is the proportion of the packets from flow j in the history, and the sum of proportions from all the flows should be 1.

The assumption that there is always a single packet in a time-slot is unrealistic. We can easily modify the estimator to account for multiple packets in a time-slot. Here we give the modified estimator, without giving the details of its derivation, which is presented in [1]. Let there be n active flows and each flow can send a packet in a time slot with a probability p. Therefore the number of packets in a time-slot is binomially distributed with parameters (n, p). The modified estimator will take a form as follows

$$H_j(t) = H_j(t-1)\left(1-\frac{np}{T}\right) + \frac{\tilde{\Psi}_j(t)}{T}, \quad (11)$$

where

$$\tilde{\Psi}_j(t) = \begin{cases} 1 & \text{if a packet from flow } j \text{ arrives in the time-slot } t \\ 0 & \text{otherwise.} \end{cases}$$

When $p = \frac{1}{n}$, we have only one packet in a time slot, which is the original case we considered. For $p = \frac{1}{n}$, (11) becomes,

$$H_j(t) = H_j(t-1)\left(1-\frac{1}{T}\right) + \frac{\tilde{\Psi}_j(t)}{T}, \quad (12)$$

which is same as (10). Extension of the estimator to variable length packets is also straightforward. Suppose the minimum packet size be m units. When a packet of size l ($l > m$) arrives, $H_j(.)$ should be updated as follows

$$\begin{aligned} H_j(t+1) &= H_j(t)\left(1-\frac{1}{T}\right)^{\left(\lfloor \frac{l}{m}+\frac{1}{2}\rfloor\right)} \\ &+ \frac{\left(\lfloor \frac{l}{m}+\frac{1}{2}\rfloor\right)\tilde{\chi}_j(t)}{T} \end{aligned} \quad (13)$$

As an illustration of the effectiveness of the above estimator, consider three flows as shown in Figure 4. The first flow sends at a rate of 100Kbps for 300s, but increases its sending rate to 300Kbps after that. The sending rate of second and third flows are approximately 200Kbps and 400Kbps respectively. The relative sending rates of the three flows until 300s are 0.142 $\left(\frac{100}{100+200+400}\right)$, 0.285 $\left(\frac{200}{100+200+400}\right)$, and 0.571 $\left(\frac{400}{100+200+400}\right)$ respectively. After 300s, flow 1 changes its sending rate, which in turn changes the relative sending rate of all the three flows. The relative sending rate of all the flows after 300s is 0.333 $\left(\frac{300}{900}\right)$, 0.222 $\left(\frac{200}{900}\right)$, and 0.444 $\left(\frac{400}{900}\right)$ respectively. We can clearly see that estimator estimates the relative rates very well. Also the estimator adapts quickly to the changes in the sending rate as given by the sudden increase in the sending rate of the first flow at time = 300s.

In the following two sections we describe an algorithm called Fair Sending Rate Estimate Based Queue Management Scheme (FSREQM), which uses the estimator $H_j(\cdot)$ to ensure fairness among flows.

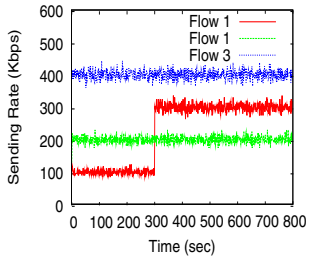

Fig. 4. Sending rates of three flows

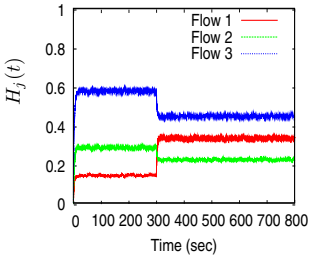

Fig. 5. Estimation of relative sending rates of flows when $T = 400$

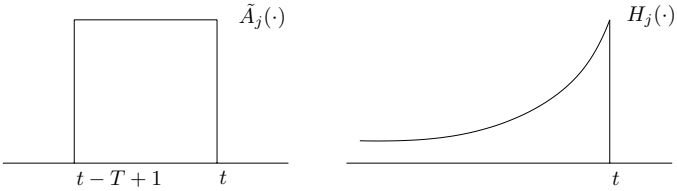

Fig. 6. Averaging procedure for $\tilde{A}_j(\cdot)$ and $H_j(\cdot)$

III. Design of Algorithm to Prevent Bandwidth Abuse

From the previous section we know that $H_j(\cdot)$ estimates the relative sending rate of flow j. A straightforward approach to prevent bandwidth abuse is to devise an algorithm which drops packets from a flow if relative share of that flow exceeds a certain predetermined value. Let K be the relative fair share

of a flow, then the dropping probability will then have a form given by

$$P_j(t) = \begin{cases} 0 & H_j(t) \leq K \\ 1 & H_j(t) > K \end{cases}$$

The relative fair share of each flow changes with changes in network conditions such as changes in the number of active flows and changes in the sending rate of active flows. Therefore, we cannot fix the value of K. We start with an initial arbitrary value of K and then change K dynamically based on the level of congestion. We use the instantaneous queue size as an indicator for the level of congestion. Specifically, when the queue size goes below a threshold, q_{min}, we increase K and when the queue size goes above a threshold, q_{max}, we decrease K.

In the relative sending rate estimation procedure, when a packet arrives, we increment the $H_j(t)$ of the flow from which the packet arrives by $\frac{1}{T}$ and then decrement each of the $H_j(t)$ by $\frac{H_j(t)}{T}$. The net reduction from all the $H_j(t)$s is $\frac{1}{T}\sum_{\forall j} H_j(t) = \frac{1}{T}$. The total reduction is same as total addition, and therefore $\sum_{\forall j} H_j(t) = 1$ holds all the time.

To make various quantities integers, we can multiply everything by T. If we do that, the total addition and total reduction when a packet arrives would be 1 and total number of packets in history would become T.

In order to prevent updating all the $H_j(t)$ when a packet arrives, we can pick one of the $H_j(t)$s with a probability proportional to $H_j(t)$ and then decrement it by the total reduction from the previous algorithm, which is 1. This method is not exactly same as the method for computing $H_j(\cdot)$, but on an average would provide a similar performance as the previous method, but with much lower computational complexity. We denote the new estimator by $S[i]$ for flow i.

The entire procedure is depicted in Algorithm 1. The algorithm maintains a vector, S, with an entry in the vector for each flow, $S[i]$ being the entry for flow i. When a packet arrives and if the sum of elements (entries) in the vector is less than T, then the element in the vector corresponding to the flow from which the packet arrived is incremented by one. Instead if if the sum of elements in the vector is greater than or equal to T, then one of the elements in the vector S is picked with a probability proportional to the value of that element. The picked entry is subtracted by one and the entry in the vector corresponding to the flow from which the packet arrived is incremented by one (lines 1-6). The amount added and subtracted are equal and therefore $\sum_{\forall i} S[i] = T$ will be true in the steady state. If the value of $S[i]$ corresponding to the flow from which the packet arrived is less than or equal to K, the packet is added, else the packet is dropped (lines 7-11). We change the value of K dynamically to reflect the changes in the characteristics of the incoming traffic as well as the level of congestion. This change is governed by the current queue size. If the queue size is larger than some maximum threshold q_{max}, which is an indication of congestion, the value of K is decreased by one. This results in restricting the sending rate of flows. Likewise, when the current queue size is below some minimum threshold q_{min}, which is an indication of low link utilization, the value of K is decreased by one, allowing flows to come in at a faster rate. To ensure a smooth variation of K, the update procedure of K is done once in F packet arrivals (lines 15-22). The parameter F is called the congestion parameter and is a representative of how fast the system responds to congestion.Other possible variations of the above algorithm are presented in [10].

Algorithm 1 FSREQM :: onPacketArrival(packet P)

```
1:  if (∑_∀i S[i] ≥ T) then
2:      pick an element S[i] from the vector S with a probability proportional to S[i].
3:      S[i]−−;
4:  end if
5:  x ⇐ flow id of packet P
6:  S[x] ⇐ S[x]++
7:  if (S[x] ≤ K) then
8:      add packet P to the queue
9:  else
10:     drop packet P
11: end if
12: if (count > 0) then
13:     count−−;
14: else
15:     if (queue size < q_min) then
16:         K++;
17:         count ⇐ F;
18:     end if
19:     if (queue size > q_max); then
20:         K−−;
21:         count ⇐ F;
22:     end if
23: end if
```

FSREQM does not require precise parameter settings. There are mainly three parameters, the history parameter T, the fair share parameter K and the congestion parameter F. If T is small, the variance of the estimator is large and if T is large then estimator takes a long time to estimate the relative sending rate correctly. In all our simulations, we assume a value of $T = 400$, which is shown to perform well [1]. The initial value assigned to K does not have any serious impact on the performance as K is changed dynamically based on the level of congestion. In our simulations we assigned the initial value of K as $\left(\frac{T}{\text{expected number of flows}}\right)$. The congestion parameter, F, is the most difficult parameter to set. The parameter F determines how fast we change the value of the fair share parameter. For a smooth variation of K, we chose a value of 200 for the parameter F.

IV. Experimental Results

In order to demonstrate the effectiveness of our scheme in achieving fairness, we perform several simulation studies using *ns-2* simulator. We conduct two different set of experiments,

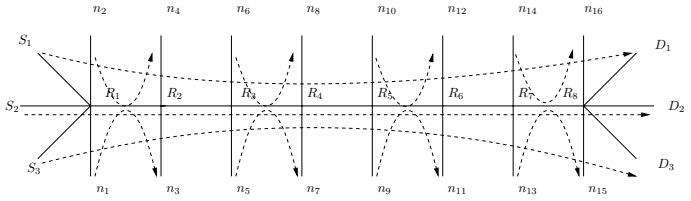

Fig. 7. Parking Lot Topology

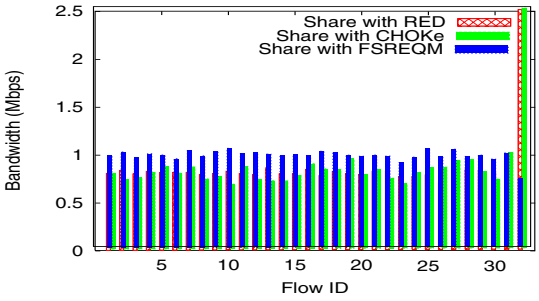

Fig. 8. The bandwidth achieved by 32 different flows with RED, CHOKe and FSREQM. Flows 31 and 32 are UDP flows sending at 1Mbps and 7Mbps respectively. With FSREQM, the UDP flow sending at 7Mbps is restricted so that other legitimate flows can get a higher bandwidth.

one using a single congested link to compare FSREQM with some other popular queue management schemes such as the RED and the CHOKe, and the second set of experiments to show that FSREQM achieve an approximate max-min fairness [1].

In the first set of experiments we assume a dumbbell topology with a single bottleneck link. There are 32 links each of bandwidth 10Mbps, while the bottleneck link is of bandwidth 32Mbps. There are 32 flows of which 30 are TCP flows and two are UDP flows. One of the UDP flow is a malicious flow sending at a very high rate of 7Mbps, whereas the other UDP flow is a legitimate UDP flow sending at a rate 1Mbps. Figure 8 shows the bandwidth achieved by each flow with RED, CHOKe and FSREQM. With both RED and CHOKe, the bandwidth received by the malicious UDP flow is close to 7Mbps and therefore the bandwidth achieved by the TCP flows are around 0.75Mbps. With FSREQM, the malicious UDP flow is restricted and the TCP flows receive a much higher bandwidth. Also, the link utilization achieved by FSREQM is very close to 100 %.

In the second case, we consider a parking-lot topology as shown in Figure 7. The flows between $(S_1, D_1), (S_2, D_2)$ and (S_3, D_3) are TCP flows and all other flows are UDP flows. The Table I shows the throughput received by each flow on the bottleneck link between R_7 and R_8 The flows between (n_2, n_4), (n_6, n_8), (n_{10}, n_{12}), and (n_{14}, n_{16}) are UDP flows sending at a rate of 1Mbps. The flows between (n_1, n_3), (n_5, n_7), (n_9, n_{11}), and (n_{13}, n_{15}) are also UDP flows, but sending at a higher rate of 5Mbps. All the bottleneck links (links between R_i and R_j) have a capacity of 10Mbps, which means that the first UDP flow is sending at a higher rate than its fair share and the second UDP flow is sending

TABLE I

MAX-MIN FAIRNESS ACHIEVED BY FSREQM

Flow id.	Average Throughput (Mbps)	Max-min fair share
1 (TCP)	2.111	2.25
2 (TCP)	2.307	2.25
3 (TCP)	2.249	2.25
4 (UDP - 5Mbps)	2.152	2.25
5 (UDP - 1Mbps)	0.984	1.0

at a rate lower than its fair share. We can see that the bandwidth allocation by FSERQM is very close to the max-min bandwidth allocation.

These simulations show that FSREQM can successfully prevent bandwidth abuse at the router, while keeping the link utilization very high. For achieving precise max-min fairness of flows parameter tuning is required. While it is easy to configure the parameters to achieve max-min fairness, there is no single set of parameters that will achieve max-min fairness for all considered cases.

V. CONCLUSION AND FUTURE WORK

In this paper, the problem of preventing bandwidth abuse is addressed. Rigorous mathematical interpretation of a class of queue management schemes based on estimating the relative sending rate of flows is presented and their potential application to preventing bandwidth abuse is discussed. A highly efficient, simple and scalable algorithm called the FSREQM is presented and its effectiveness in preventing bandwidth abuse is shown by extensive simulations. Future work involves testing the system using real traffic traces and evaluating the performance of the algorithm in real, operational networks.

REFERENCES

[1] V. Ramaswamy, "Efficient Control of Non-Cooperative Traffic Using Sending Rate Estimate-Based Queue Management Schemes," Ph.D. dissertation, The University of Mississippi, 2006.

[2] W. R. Stevens, *TCP/IP Illustrated.* Addison-Wesley Professional, December 1993, vol. 1.

[3] Z. Zhao, J. Ametha, D. Swaroop, and A. Reddy, "A Method for Estimating Non-Responsive Traffic at a Router," in *Proc. of the ACM Conference on SIGMETRICS'02*, June 2002, pp. 274–275.

[4] I. Stoica, S. Shenker, and H. Zhang, "Core-stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocation in High Speed Networks," in *Proc. of ACM SIGCOMM'98*, Sep 1998, pp. 118–130.

[5] R. Pan, B. Prabhakar, and K. Psounis, "CHOKe, A Stateless Active Queue Management Scheme for Approximating Fair Bandwidth Allocation," in *Proc. of IEEE INFOCOM'00*, July 2000, pp. 942–951.

[6] R. Pan, L. Breslau, B. Prabhakar, and S. Shenker, "Approximate Fairness Through Differential Dropping," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 2, pp. 23–39, 2003.

[7] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker, "On the Characteristics and Origins of Internet Flow Rates," in *Proceedings of ACM SIGCOMM'02*, August 2002, pp. 161–174.

[8] W. Feng, D. Kandlur, D. Saha, and K. Shin, "Stochastic Fair BLUE: A Queue Management Algorithm for Enforcing Fairness," in *Proceedings of IEEE INFOCOM'01*, April 2001, pp. 1520–1529.

[9] P. McKenny, "Stochastic Fairness Queueing," in *Proceedings of IEEE INFOCOM'90*, June 1990, pp. 733–740.

[10] V. Ramaswamy et al., "Light-Weight Control of Non-Responsive Traffic with Low Buffer Requirements," in *Proc. of IFIP Networking'07*, May 2007.